Constraint Programming Formulation for the Elevator Trip Origin-Destination Matrix Estimation Problem

Juha-Matti Kuusinen, Arnaud Malapert

LABORATOIRE

INFORMATIQUE, SIGNAUX ET SYSTÈMES
DE SOPHIA ANTIPOLIS
UMR7271

Constraint Programming Formulation for the Elevator Trip Origin-Destination Matrix Estimation Problem

Juha-Matti Kuusinen, Arnaud Malapert

Équipe CeP

Rapport de Recherche
ISRN I3S/RR-2014-01-FR

Janvier 2014

Constraint Programming Formulation for the Elevator Trip Origin-Destination Matrix Estimation Problem

Juha-Matti Kuusinen [1] , Arnaud Malapert [2],

Équipe CeP

ISRN I3S/RR-2014-01-FR
Janvier 2014 - 16 pages

Abstract: We present a constraint programming formulation for the elevator trip origin-destination matrix estimation problem, and propose different approaches to solve the problem. An elevator trip consists of successive stops in one direction of travel with passengers inside the elevator. It can be defined as a directed network, where the nodes correspond to the stops on the trip, and the arcs to the possible origins and destinations of the passengers boarding and alighting at the stops.
The goal is to estimate the count of passengers for the origin-destination pairs of every elevator trip occurring in a building. These counts are used to make passenger traffic forecasts which are needed in elevator dispatching to make robust dispatching decisions in constantly changing traffic conditions. The proposed approach can be used to estimate elevator trip origin-destination matrices within a real time limit, and provides a flexible method to solve a challenging real life problem.

Key-words: Elevator Traffic ; Origin-Destination Matrix ; Constraint Programming

[1] KONE Corporation, POB 7, 02151 Espoo, Finland – juha-matti.kuusinen@kone.com
[2] Univ. Nice Sophia Antipolis, CNRS, I3S, UMR 7271, 06900 Sophia Antipolis, France – arnaud.malaert@unice.fr

Constraint Programming Formulation for the Elevator Trip Origin-Destination Matrix Estimation Problem

Juha-Matti Kuusinen[1] and Arnaud Malapert[2]

[1] KONE Corporation, POB 7, 02151 Espoo, Finland
juha-matti.kuusinen@kone.com
[2] Universite Nice Sophia Antipolis, I3S UNS CNRS, 06903 Sophia Antipolis, France
arnaud.malapert@unice.fr

Abstract. We present a constraint programming formulation for the elevator trip origin-destination matrix estimation problem, and propose different approaches to solve the problem. An elevator trip consists of successive stops in one direction of travel with passengers inside the elevator. It can be defined as a directed network, where the nodes correspond to the stops on the trip, and the arcs to the possible origins and destinations of the passengers boarding and alighting at the stops. The goal is to estimate the count of passengers for the origin-destination pairs of every elevator trip occurring in a building. These counts are used to make passenger traffic forecasts which are needed in elevator dispatching to make robust dispatching decisions in constantly changing traffic conditions. The proposed approach can be used to estimate elevator trip origin-destination matrices within a real time limit, and provides a flexible method to solve a challenging real life problem.

Keywords: elevator traffic, origin-destination matrix, constraint programming

1 Introduction

Modern elevator systems consist of one or more elevator groups. Each group includes typically four to eight elevators that share the same call giving devices in the elevator lobbies, and that are controlled collectively. The main task of the control unit is to allocate passenger calls to elevators by minimizing, e.g., passenger waiting time. These elevator dispatching decisions define the routes of the elevators. Modern group controls plan the elevator routes based on existing calls [1, 2]. At any given moment, however, a passenger may arrive to an elevator lobby and give a new call which requires the changing of previously defined routes, if they are no longer optimal. By making forecasts of future passengers, the group control can avoid such unexpected route changes and improve passenger service level. The forecasts should be based on the realized passenger journeys between every pair of floors of a building. The problem is that, especially during heavy traffic, the passenger journeys cannot be uniquely determined.

The passenger journeys can, however, be estimated by solving the elevator trip origin-destination matrix (ETODM) estimation problem [3]. An elevator trip to up or down direction starts when passengers board an empty elevator and ends to a stop where the elevator becomes empty again. The passengers who board the elevator register calls that define their destinations, and the OD pairs of the trip. The boarding and alighting passenger counts can be measured, e.g., with an electronic load weighing device [4]. If the counts are measured without errors, they are consistent. The ETODMs estimated for a given time interval can be combined into a building OD matrix (BODM) that describes the passenger traffic between every pair of floors in the building during that interval. The BODMs can be used to make forecasts about future passengers. One method to construct BODMs from consistent counts was presented in [5]. Another method that accepts inconsistent counts was presented in [3]. In this paper, we present a new approach for constructing BODMs from consistent and inconsistent counts.

An elevator trip is analogous to a single transit route, e.g., a bus line, where there is only one route connecting any OD pair, and usually counts of the boarding and alighting passengers are collected on all stops on the route [6]. There are many methods for estimating the OD matrix for a single transit route [6–17]. If the observed passenger counts are consistent, then a typical objective is to minimize a distance measure between the unobserved and a target OD matrix subject to the flow conservation constraint. This constraint simply requires that passengers travelling on the route do not disappear or multiply. The target OD matrix is usually based on historical data or a survey. If the counts are not consistent, then a distance measure between the unobserved and observed counts should also be minimized.

A single transit route is usually defined in advance and remain as such for long periods of time. This means that it is possible to collect many counts on the same route during a given time period, e.g., a rush hour, and use these counts to estimate average passenger counts for the OD pairs of the route. An elevator trip is request driven, which means that there may not be two similar elevator trips even within a day. Hence, it may be impossible to make many observations on the same trip. In addition, every elevator trip has its own set of OD pairs, and boarding and alighting counts. This is why we need to estimate a separate OD matrix for each elevator trip. Because there cannot be partial passengers, only integer solutions are acceptable.

We formulate the ETODM estimation problem as a constraint optimization problem (COP) [18]. In addition to respecting a set of constraints defined by the formulation, a solution to the problem is optimal with respect to a predefined distance measure between the unobserved and observed passenger counts. We selected the least squares (LS) objective function for the following reason. Experience has shown that usually, in case an error occurs, the measured boarding or alighting count is usually one passenger less than the true count independent of the number of passengers who board or alight the elevator. This is because a typical passenger elevator door width is at most 1200 millimeters, and thus,

usually at most two passengers may board or alight at the same time and be measured as one. The LS objective does not allow large deviations between the unobserved and observed counts, and thus, is a reasonable choice.

Our approach is studied with respect to solving time and BODM quality. Solving time is used since, for implementing an ETODM estimation algorithm in a real elevator group control application, the algorithm must be fast to reduce the computational load of the group control computer, and to have the most recent information about the passenger traffic all the time. BODM quality affects the robustness of the passenger traffic forecasts. It is measured based on the total squared deviation, accuracy and precision.

Kuusinen et al. [3] formulated the ETODM estimation problem as a box-constrained integer least squares (BILS) problem, and presented algorithms for finding all solutions to the problem. When all solutions are available and one is selected randomly every time, the BODMs are not affected by the algorithm used in solving the problem. In the long term, this strategy results to BODMs that model better all possible realizations of the passenger traffic, and enable robust passenger traffic forecasting in elevator dispatching. However, from a statistical point of view, finding all solutions and then selecting one randomly is equal to finding a single solution randomly. Intuitively, the latter is a faster strategy. One advantage of the CP approach compared to, e.g., mixed integer linear programming, is that an optimization procedure resulting in a single or multiple deterministic or randomized optimal solutions can be easily implemented. Another advantage is that an existing CP algorithm can be directly applied with a reasonable amount of additional work.

The rest of the paper is organized as follows. Section 2 presents the CP formulation for the ETODM estimation problem, and Sec. 3 search algorithms for solving the problem. In Sec. 4, we describe numerical experiments to evaluate the proposed approach, and in Sec. 5 the experimental results. Section 6 concludes the paper.

2 Constraint Programming Formulation

Adapting to the notation in [3], we define an elevator trip as a directed network of nodes $N = \{1, 2, \ldots, n\}$, and arcs A defined by OD pairs (i, j), $i, j \in N$. The node i corresponds to the i-th stop on the elevator trip. Let r_i be the node at which a delivery request to the node $i \in N$, $r_i < i$, is registered. If no delivery requests are registered to node i, then $r_i = n + 1$. Let b_i and a_i denote the measured count of passengers who board and alight at node $i \in N$, respectively. The elevator capacity, expressed as number of passengers, is denoted with C.

We assume that:

1. At any time, there are less than C passengers in the elevator
2. At least one passenger boards at node $r_i \neq n + 1$ and alights at node i
3. Passengers do not alight at a node without a delivery request
4. A passenger who boards at node $j < r_i$, i.e., before the delivery request to node i is registered, does not alight at node i

The fourth assumption means that the possible destinations of a passenger are defined by the delivery requests that are registered before or at the node where the passenger boards the elevator, which is usually the case in practice. This eliminates some OD pairs, and thus, an elevator trip often includes a smaller number of OD pairs than a single transit route where typically any node i forms an OD pair with any other node j, $i < j$.

The set of arcs A is defined as:

$$A = \{(i,j) \in N^2 \mid i < j \ \wedge \ i \geq r_j\} \ . \tag{1}$$

Let $B_i \in [0,C]$ and $A_i \in [0,C]$ denote the unobserved count of passengers who board and alight the elevator at node $i \in N$, respectively. Let $P_i \in [0,C]$, $i = 1,\ldots,n-1$, denote the number of passengers in the elevator between the nodes i and $i+1$. Finally, let $X_{ij} \in [0,C]$ denote the unobserved passenger count along the arc or OD pair $(i,j) \in A$, i.e., the passenger count from origin i to destination j, that we want to estimate.

The unobserved boarding and alighting counts must be consistent:

$$\sum_{i \in N} B_i = \sum_{j \in N} A_j \ . \tag{2}$$

At the first node, the unobserved boarding count must be at least one and the alighting count zero, and at the last node the reverse must hold:

$$A_1 = 0, \quad B_1 \geq 1, \quad A_n \geq 1, \quad B_n = 0 \ . \tag{3}$$

At every node between the first and the last node, at least one passenger either boards or alights:

$$A_i + B_i \geq 1 \quad \text{if } 1 < i < n \ .$$

This constraint is more accurately stated by considering the delivery requests.

Passengers cannot alight at a node to which there is no delivery request, and thus, at least one passenger must board:

$$r_i = n+1 \Leftrightarrow A_i = 0 \wedge B_i \geq 1 \quad \text{if } 1 < i \leq n \ . \tag{4}$$

At least one passenger boards at node $r_i \neq n+1$ and alights at node i:

$$r_i \neq n+1 \Leftrightarrow X_{r_i,i} \geq 1 \quad \text{if } 1 < i \leq n \ . \tag{5}$$

The unobserved OD passenger counts are related to the unobserved boarding and alighting counts through the so called flow conservation constraints :

$$\sum_{j \mid (i,j) \in A} X_{ij} = B_i \quad \forall i \in N \ , \tag{6}$$

$$\sum_{i \mid (i,j) \in A} X_{ij} = A_j \quad \forall j \in N \ . \tag{7}$$

The number of passengers in the elevator between nodes i and $i+1$, P_i, is computed as follows:

$$P_1 = B_1, \quad P_{n-1} = A_n, \quad P_i = P_{i-1} + B_i - A_i \quad \text{if } 1 < i < n-1 \ . \tag{8}$$

The elevator capacity is always respected because of the domain of the variables.

The problem of finding the passenger counts for the arcs or OD pairs of an elevator trip such that the unobserved boarding and alighting counts are as close as possible to the measured counts can be seen as a network flow problem. In such a problem, the objective function is typically linear. A linear objective function may, however, result to a solution that produces small deviations between most of the unobserved and observed counts, but accepts large deviations for some counts. This is not good since, as explained before, the observed boarding or alighting count is usually one passenger less than the true count. Hence, we consider the LS deviation between the unobserved and observed counts as the objective function:

$$\sum_{i \in N} \left[(A_i - a_i)^2 + (B_i - b_i)^2 \right] \ . \tag{9}$$

The goal is to minimize (9) with respect to the constraints (2)-(5) and (6)-(8).

Note that the LS objective value in (9) is zero only if the problem is consistent. This is the case if:

$$b_i = \sum_{j \in N} a_j \ , \quad b_i \geq \sum_{j \,|\, r_j = i, i < j \leq n} 1 \quad \forall i \in N \ ,$$

$$a_j \leq \sum_{i \,|\, 1 \leq i < j} \left(b_i - \sum_{k \,|\, r_k = i, i < k \leq n, k \neq j} 1 \right) \quad \forall j \in N \ .$$

Similar consistency conditions were defined in [3].

3 Search Algorithms

We consider a complete standard backtracking search which consists of a depth-first traversal of the search tree. At a node of the search tree, an uninstantiated variable is selected and the node is extended so that the resulting new branches out of the node represent alternative choices that may have to be examined in order to find a solution. The branching strategy determines the next variable to be instantiated, and the order in which the values from its domain are selected.

During a depth-first search, constraint propagation is used to remove inconsistencies. This can make the branching strategy more efficient. Such as the standard depth-first branch-and-bound algorithm for solving a discrete optimization problem, our algorithm performs a backtracking search and maintains a lower bound, lb, and an upper bound, ub, on the objective value. When $ub \leq lb$, the sub tree can be pruned because it cannot contain a better solution. The efficiency of the algorithm depends strongly on its pruning capacity, which relies on the quality of the bounds.

3.1 Branching Strategy

A branching strategy determines the next variable to be instantiated (variable selection), and the next value the variable is assigned from its current domain (value selection). The branching strategy strongly impacts the performance of the search by improving the detection of solutions (or failures for unsatisfiable problems) when building the search tree.

Here we consider the following variable selection strategies: dom selects the variable whose domain is minimal; wdeg selects the variable whose weighted degree is maximal; lex selects a variable according to lexicographic ordering; dom/wdeg selects the variable that minimizes the quotient of its domain size over its weighted degree; random selects a variable randomly. The weighted degree of a variable is the sum of the weights of the constraints in the current search state associated with the variable. The weight of a constraint is incremented by one each time it causes a failure during the search. We consider only two classical value selection strategies: minVal selects the smallest value randVal selects a value randomly. There is also a third classical value selection strategy, maxVal, which selects the largest value. However, our numerical experiments indicated that it is less efficient than minVal, and thus, is not considered in this study.

3.2 Optimization Procedure and Candidate Algorithms

Most CP tools use by default the top-down procedure. It starts with an upper bound ub which is updated whenever a better solution is found. Here, the problem is solved using the bottom-up procedure. The procedure starts with a lower bound lb as a target upper bound which is incremented by one unit until the problem becomes feasible. The first solution found by the bottom-up procedure is proven optimal. If (by luck) the first solution found by the top-down procedure is optimal, the optimality has to be still proven. Most bottom-up variants differ by the way infeasibilities are resolved.

Let opt denote the optimal objective value. The bottom-up procedure solves opt − lb unsatisfiable problems and only one satisfiable problem before finding an optimal solution. Hence, the number of problems that has to be solved is linear with respect to lb. Most bottom-up variants reduce from a linear to a logarithmic number of iterations in the worst-case. The bottom-up procedure is efficient if the difference between the lower bound and the optimum is tight, whereas other procedures, such as the top-down, are useful if the difference is large, or when the goal is to find good solutions quickly. In our case, the boarding and alighting counts are often measured without errors, and if an error is made, it rarely exceeds one unit. This means that the optimal objective value is often equal or close to zero, and thus, the bottom-up procedure with the initial lower bound equal to zero, $lb = 0$, is a good candidate.

The candidate algorithms (CA) are defined in Tab. 1. Each column describes the alternatives for a component of the backtrack search. A CA is obtained by combining one variable selection and one value selection strategy. For example, DM uses dom for variable selection and minVal for value selection.

Table 1. Candidate algorithms

Variable selection	Value selection
D – dom	
W – dom/ wdeg	M – minVal
L – lex	R – randVal
R – random	

4 Numerical Experiments

4.1 Simulation Process

We simulated lunch hour traffic in a 25 floors high office building using the Building Traffic Simulator (BTS) [19]. Lunch hour is one of the most difficult traffic situations occurring in a building during a day. The simulation time was 15 minutes which is a typical interval length for passenger traffic statistics in a real elevator group control application [20]. In a simulation, a specific traffic pattern such as lunch hour traffic is defined by traffic components which are incoming, outgoing and inter-floor traffic. Incoming component consists of passengers who travel from entrance floors to upper floors, outgoing of passengers who travel from upper floors to entrance floors, and inter-floor of passengers who travel between upper floors. In a typical lunch hour traffic pattern, which was used also in this study, the proportion of incoming, outgoing and inter-floor traffic is 40%, 40% and 20%, respectively.

We used a group of eight elevators with the capacity of 21 passengers, and adjusted the traffic intensity so that the handling capacity (HC) of the elevator group was insufficient. HC is the number of passengers the elevator group can transport within a five minutes period during up-peak [21]. Up-peak consists only of incoming passengers. When the HC is insufficient, the elevators become often fully loaded, and thus, make many stops during one up or down trip. This increases the number of difficult problem instances. Because, in practice, elevator groups are designed to have enough HC, the problems occurring in reality are likely to be less complex.

Every simulation produces a log file that contains all the data related to the simulation, e.g., all passengers and their origin and destination floors. From this data, we can construct the source BODM. An element in the source BODM corresponds the true number of passengers from an origin to a destination. The data in the log file can also be used to construct the ETODM estimation problem instances. By solving all the ETODM estimation problem instances, and adding up the estimation results, we obtain the estimated BODM.

To obtain several sets of test data, we repeated the simulation 10 times with different seeds. Because a simulation does not involve measuring errors, the resulting 10 sets of problem instances contain only consistent instances. In practice, however, a measured boarding or alighting count may be typically one passenger less than the true count, and the frequency of the errors depends on

the accuracy of the measuring device. Experience has shown that a reasonable assumption about the measurement accuracy is 90%. Consequently, we created inconsistent problem instances from the consistent problem instances by removing one passenger from each boarding and alighting count with 10% probability. This resulted in 10 new sets of problem instances containing in total 165 inconsistent instances. To make it clear, from now on we call all of the instances in the 10 new sets inconsistent even if some of them are consistent.

4.2 Performance Measures

The quality of the estimated BODM is evaluated based on the total squared deviation, accuracy and precision. Let X_{ij}^{true} and X_{ij}^{est} denote the true and the estimated passenger count from origin i to destination j in the source and the estimated BODM, respectively, and let N denote the total number of OD pairs in the building. The total LS deviation is the sum of the OD passenger count deviations between the estimated and the source BODM:

$$\text{total squared deviation} = \sum_{i \in N} \sum_{j \in N} (X_{ij}^{est} - X_{ij}^{true})^2 \ . \tag{10}$$

Hence, the total squared deviation measures the proximity of the estimated BODM to the source BODM with respect to the OD passenger counts.

Accuracy and precision are computed as follows. First, we determine the number of true positives, false positives, and true negatives. An OD pair is classified as true positive, if $X_{ij}^{est} > 0$ and $X_{ij}^{true} > 0$, as false positive if $X_{ij}^{est} > 0$ and $X_{ij}^{true} = 0$, and as true negative if $X_{ij}^{est} = X_{ij}^{true} = 0$. Then, accuracy is defined as:

$$\text{accuracy} = \frac{\text{number of true positives} + \text{number of true negatives}}{N} \ , \tag{11}$$

and precision as:

$$\text{precision} = \frac{\text{number of true positives}}{\text{number of true positives} + \text{number of false positives}} \ . \tag{12}$$

Note that if an OD pair is classified as false positive, then the OD pair is included in at least one of the ETODM estimation problem instances and assigned a positive OD passenger count even if in the simulation no passengers travelled from the origin to the destination of the OD pair. Hence, the number of false positives is related to the magnitude of error caused by the formulation of the ETODM estimation problem. In other words, accuracy is the degree to which the problem instances obtained using the formulation, and thus, the estimated BODM, contain the true OD pairs described by the source BODM, and precision the variability of the estimation results due to the formulation.

The solving time of each problem instance should be as short as possible to have all the time the most recent information about the passenger traffic. In addition, the computer used in a real application performs simultaneously

many tasks, and the more tasks, the less time for solving an ETODM estimation problem instance. A reasonable time limit is 0.5 seconds especially during heavy traffic when the computation load is higher. This criterion was used also in [3, 5]. To compare the quality of the BODMS obtained with different CAs, the final solution to an ETODM estimation problem instance is selected as the average of the optimal solutions to the instance.

5 Experimental Results

All the experiments were conducted on a Linux machine with 32 GB of RAM and a Intel Core i7 processor (6 cores – 3.20GHz). The implementation is based on choco (http://choco.mines-nantes.fr). The industrial computer used in a real application is typically less efficient than the one used in this study. However, the solving times reported here are also longer than they would be in a real application since the choco solver is a general CP solver and therefore performs some tasks that are not necessary for the ETODM estimation problem. A solver designed only for this problem would be more efficient. Hence, the solving times reported here are somewhat comparable to those that would be obtained in reality.

5.1 Number of Optimal Solutions

Table 2 shows the distribution of the number of optimal solutions among the 558 consistent and inconsistent problem instances. For example, the fourth column indicates that there are 35 instances among the consistent instances, and 75 instances among the inconsistent instances that have between 10 and 100 optimal solutions. The last column gives the total number of optimal solutions. It can be concluded that the search space is significantly larger for the inconsistent instances, i.e., they are harder to solve.

Table 2. Distribution of the number of optimal solutions

#solutions	$= 1$	≤ 10	$\leq 10^2$	$\leq 10^3$	$\leq 10^4$	$> 10^4$	Total
Consistent	405	84	35	19	13	2	136280
Inconsistent	320	107	75	27	17	12	34063787

Table 3 gives the distribution of the LS objective value at the optimal solutions to the inconsistent problem instances. The distribution shows that the lower bound of an inconsistent instance rarely exceeds one, which confirms that the bottom-up procedure with the initial lower bound equal to zero, $\mathsf{lb} = 0$, is a good choice for optimization.

Table 3. Distribution of the LS objective value for the inconsistent instances

Objective	0	1	2	3
Frequency	393	136	27	2

5.2 Impact of Finding Multiple Solutions

In this section, we consider the deterministic algorithms DM, WM and LM, and the first $10^{\mathtt{K}}$, $\mathtt{K} = 0, 1 \ldots, 3, \star$, optimal solutions. The star sign refers to all optimal solutions. The goal is to study how the number of optimal solutions affect solving time and quality, and the differences between the deterministic CAs with respect to these performance measures.

Solving Time. Table 4 gives the solving time of each CA for the 558 consistent and inconsistent problem instances. Consistent instances are easy to solve, the total solving time being less than 74 seconds considering all CAs. The fastest CA is DM. Inconsistent instances are harder to solve. The slowest CA is LM, and DM clearly outperforms WM when all optimal solutions has to be found, although WM is slightly faster than DM for $\mathtt{K} = 0, 1, \ldots, 3$. An interesting result is that most of the time for finding all solutions to the inconsistent instances is taken only by one instance. The reason is that it has about 75 times more optimal solutions than the second hardest inconsistent instance, and the more solutions the more time it takes to find them all.

Table 4. Solving time of each CA in seconds

	Consistent			Inconsistent		
K	DM	WM	LM	DM	WM	LM
0	66.1	68.2	66.7	79.0	78.1	144.5
1	66.6	68.8	67.2	81.7	80.5	146.6
2	68.0	70.1	68.7	86.2	85.0	152.3
3	69.3	71.4	69.9	91.3	90.0	158.2
$\star$	71.3	73.5	71.9	1122.5	3392.0	3919.8

Figure 1 shows the frequency of solving time per instance for the inconsistent instances. Solving times for consistent instances are not shown since the differences between the CAs were negligible. Note that the bar heights are not equal since the number of instances for which the CAs found the optimal solutions in less than 0.25 seconds is not shown. The number of inconsistent instances for which the CAs cannot find the first $10^{\mathtt{K}}$, $\mathtt{K} = 1, 2$, optimal solutions within 0.5 seconds is small. For the first $10^{\mathtt{K}}$, $\mathtt{K} = 3, \star$, optimal solutions DM and WM are faster than LM, and the number of instances for which the latter two cannot find

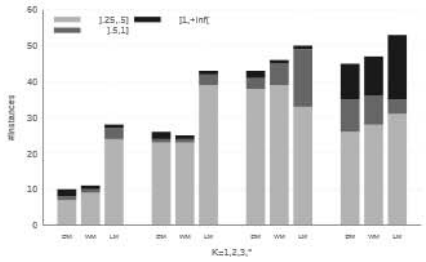
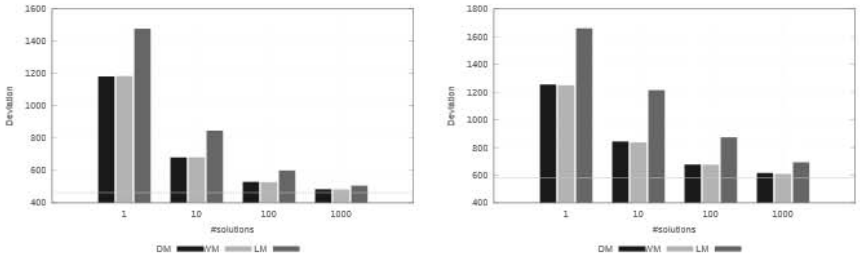


Fig. 1. Frequency of solving time for inconsistent instances

all optimal solutions within 0.5 seconds is only 19 which is about 3.4% of all the 558 instances. This result is comparable to that obtained in [3].

Total Squared Deviation. Figure 2 shows the total squared deviation of each deterministic CA for consistent (2(a)) and inconsistent (2(b)) instances as histograms. The total squared deviation for each BODM is computed based on (10), and the results shown in the figure are obtained by summing up these deviations. The CAs are grouped by the parameter K, $K = 0, 1, \ldots, 3$, and the horizontal line is the total squared deviation of the CAs for all optimal solutions, $K = *$, which is the same for all CAs.



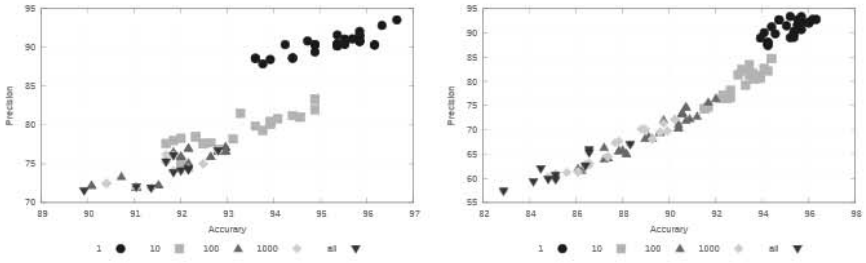
(a) Consistent (b) Inconsistent

Fig. 2. Total squared deviation

According to the figure, inconsistent instances lead to a greater deviation. The reason is that inconsistent instances are created by removing passengers from the observed counts, and thus, the total count of passengers obtained by adding up the OD passenger counts of an optimal solution is always less than the true total count of passengers. In addition, finding multiple optimal solutions reduces the deviation. This is because the final solution to a problem instance is computed as the average of the optimal solutions to the instance. This implies that the more optimal solutions, the more small errors between the true and the

estimated OD passenger counts, and the smaller the total squared deviation. A single optimal solution usually results to a few large errors which will make the total squared deviation large. LM results always to the greatest deviation, and DM and WM are almost equivalent.

Accuracy and Precision. Figure 3 shows the accuracy and precision for each consistent (3(a)) and inconsistent (3(b)) BODM and for each CA as a scatter plot. Each point represents one BODM computed by a CA for $K = 0, 1, \ldots, 3, *$, and x coordinate is its accuracy and y coordinate its precision. It seems that the more optimal solutions, the less accuracy and precision. There are two reasons for this result. First, the problem formulation of any instance may contain and OD pair for which the OD passenger count in the source BODM is zero, and thus, should not be included in any of the formulations. Second, since the final solution to an instance is selected as the average of all the optimal solutions to the instance, the more optimal solutions, the more likely that an OD pair, for which the OD passenger count in the source BODM is zero, receives a non zero value and is classified as false positive. This increases the number of false positives and decreases the number of true negatives, and thus, reduces accuracy and precision. Furthermore, inconsistent instances reduce accuracy and precision for the same reason they lead to a greater total squared deviation.



(a) Consistent (b) Inconsistent

Fig. 3. Accuracy and precision

5.3 Impact of Randomization

Previous results show that DM is better than WM and LM. It is also easier to implement in practice. Hence, we use DM to study the effect of randomization. From the randomized algorithms, we consider only DR and RR, and 10^K, $K = 0, 1, \ldots, 3$, optimal solutions. The difference between the deterministic and randomized algorithms is that the deterministic algorithms always find the first 10^K, $K = 0, 1, \ldots, 3$, optimal solutions whereas the randomized algorithms find 10^K randomized solutions among all optimal solutions.

Randomized search algorithms are usually less efficient than deterministic ones for solution enumeration, and the effect of the value selection is also reduced. Moreover, a randomized search typically gives a different set of optimal solutions per problem instance when it is solved several times. Hence, to study the robustness of the randomized algorithms, we ran DR 50 times for consistent and inconsistent instances, and RR 50 times for consistent instances, but only five times for inconsistent instances because of much longer solving times. One run consists of solving all the 558 instances corresponding to the 10 BODMs once. Hence, each run produces 10 estimated BODMs.

Solving Time. Table 5 gives the average solving time (avg) among all runs, and their standard deviation (std). The latter is shown only for the inconsistent instances since for the consistent instances the standard deviation was always below 0.05 seconds. For consistent instances, randomization does not considerably increase the solving time. For inconsistent instances, DR produces a small increase, but RR is far too slow. Again, it takes a long time to solve the hardest inconsistent instance, but also many other inconsistent instances become problematic. RR has also a greater deviation, but this is partly because the number of runs is only five.

Table 5. Solving time of each CA in seconds

	Consistent			Inconsistent				
	DM	DR	RR	DM	DR		RR	
K		avg	avg		avg	std	avg	std
0	66.1	66.7	66.7	79.0	82.6	3.7	4454.4	256.0
1	66.6	67.4	67.4	81.7	85.1	3.7	4436.0	254.6
2	68.0	68.8	68.9	86.2	88.8	3.8	4604.8	160.6
3	69.3	70.4	70.6	91.3	92.8	3.8	4440.6	219.5

According to Fig. 4, DR can find the optimal solutions in at most 0.5 seconds for about 99.5% of the inconsistent problem instances independent of the value of K. This is even a better result than for the fastest deterministic CA DM, which can find the first 10^3 optimal solutions in at most 0.5 seconds for about 99.1% of the inconsistent problem instances. RR gives comparable results for $K = 0, 1$, but for $K = 2, 3$ it becomes much slower. It can find 10^3 optimal solutions in at most 0.5 seconds for about 94% of the inconsistent problem instances.

Total Squared Deviation. Table 6 gives results on the total squared deviation of each CA. The total squared deviation of one run is the sum of the total squared deviations between the 10 estimated and source BODMs. The average (avg) and standard deviation (std) are computed over all runs. Also, the maximum (max) and the minimum (min) of all runs are shown.

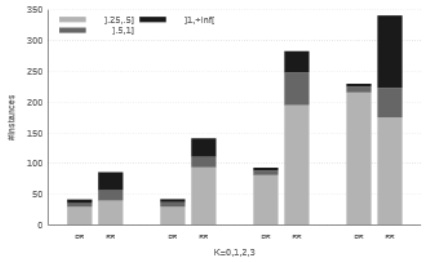


Fig. 4. Frequency of solving time for inconsistent instances

For consistent instances, both DR and RR result on average to a smaller total squared deviation than DM for all values of K. In the worst case (max), DR is better than RR, and in the best case (min) RR is better than DR except when only one optimal solution has to be found. The standard deviations of DR are slightly smaller than those of RR. For inconsistent instances, DR is on average better than DM and RR, the latter two being about equivalent. In the best case, DR is better than RR, which is true also for the worst case except when only one optimal solution has to be found. For the randomized CAs, it can be concluded that the more optimal solutions, the more stability, i.e., less variation between the estimated BODMS, and the more quality, i.e., smaller differences between the estimated and source BODMs.

Table 6. Total squared deviation

		DM	DR				RR			
	K		avg	std	min	max	avg	std	min	max
Consistent	0	1182.0	978.1	159.0	648.0	1316.0	979.0	163.2	674.0	1386.0
	1	681.5	613.3	66.3	488.4	757.1	629.2	91.5	432.8	838.3
	2	530.5	512.6	33.1	447.4	587.9	516.7	47.2	413.4	627.2
	3	484.5	477.8	14.0	453.5	499.3	475.8	19.8	436.6	525.6
Inconsistent	0	1255.0	1179.0	164.5	867.0	1575.0	1253.8	199.8	1035.0	1527.0
	1	843.7	762.1	79.9	607.8	962.0	848.0	127.9	709.1	1025.6
	2	676.9	653.4	39.5	578.1	745.5	684.3	65.5	602.6	771.1
	3	616.5	599.2	18.5	561.1	638.9	615.9	39.1	573.0	670.4

Accuracy and Precision. Based on the previous results, DR is better than RR with respect to all performance measures, and thus, we consider here only DR. The behaviour of DR with respect to accuracy and precision is similar to the deterministic CAs (see Fig. 3). Figure 5 compares the accuracy and precision of DR and DM for $K = 0, 1, \ldots, 3$. The x coordinate is the ratio of the average

accuracy of DR to the accuracy of DM, and y coordinate is the same ratio for the precision. All points located above and to the right of the point (1,1) are BODMs for which the accuracy and precision are improved by the use of DR. On the contrary, all points located below and to the left of the point (1,1) are BODMS for which the accuracy and precision are degraded by the use of DR. All points are located around the diagonal as the variation of the accuracy is roughly proportional to the variation of the precision. It can be concluded that in general DR reduces accuracy and precision.

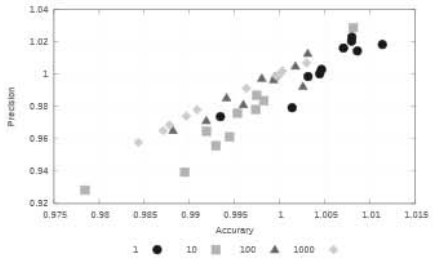


Fig. 5. Comparison of DR and DM in terms of accuracy and precision

6 Conclusion

In this paper, we presented a constraint programming (CP) formulation for a real life problem where the goal is to estimate the origin-destination (OD) passenger counts for the OD pairs of an elevator trip, i.e., the elevator trip OD matrix (ETODM). The ETODMs estimated for a given time interval can be combined into a building OD matrix (BODM) that describes the passenger traffic between every pair of floors in the building during that interval. These matrices form traffic statistics that can be used to make forecasts about future passengers. The forecasts are needed in elevator dispatching to avoid bad dispatching decisions with respect to future passengers.

We compared three deterministic and two randomized CP algorithms in finding a single, a predefined number or all optimal solutions to the ETODM estimation problem. The comparison was based on solving time and BODM quality which affects the reliability of the passenger traffic forecasts. The results suggest that randomization and multiple solutions is a good compromise between solving time, quality and robustness. In addition, the best randomized algorithms fulfill real time elevator group control requirements for solving ETODM estimation problems.

References

1. Tyni, T., Ylinen, J.: Genetic algorithms in elevator car routing problem. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001). (2001) 1413–1422
2. Koehler, J., Ottiger, D.: An AI-based approach to destination control in elevators. AI Magazine **23**(3) (2002) 59–78
3. Kuusinen, J.M., Sorsa, J., Siikonen, M.L.: The elevator trip origin-destination matrix estimation problem. Transportation Science (to appear)
4. Siikonen, M.L.: Elevator group control with artificial intelligence. Research report a67, Helsinki University of Technology, Systems Analysis Laboratory (1997)
5. Kuusinen, J.M., Ruokokoski, M., Sorsa, J., Siikonen, M.L.: Linear programming formulation of the elevator trip origin-destination matrix estimation problem. In: Proceedings of the 2nd International Conference on Operations Research and Enterprise Systems. (2013) 298–303
6. Nguyen, S.: Estimating origin-destination matrices from observed flows. In Florian, M., ed.: Transportation Planning Models, North-Holland, Amsterdam (1984) 363–380
7. van Zuylen, H., Willumsen, L.: The most likely trip matrix estimated from traffic counts. Transportation Research B **14** (1980) 281–293
8. Lamond, B., Stewart, N.: Bregman's balancing method. Transportation Research B **15**(4) (1981) 239–248
9. Bell, M.: The estimation of an origin-destination matrix from traffic counts. Transportation Sci. **17**(2) (1983) 198–217
10. Ben-Akiva, M., Macke, P., Hsu, P.: Alternative methods to estimate route-level trip tables and expand on-board surveys. Transportation Research Record **1037** (1985) 1–11
11. Tsygalnitsky, S.: Simplified methods for transportation planning. Master's thesis, Department of Civil Engineering, Massachusetts Institute of Technology, Cambridge (1977)
12. Li, Y., Cassidy, M.: A generalized and efficient algorithm for estimating transit route ods from passenger counts. Transportation Research B **41**(1) (2007) 114–125
13. Li, B.: Markov models for bayesian analysis about transit route origin-destination matrices. Transportation Research B **43**(3) (2009) 301–310
14. Spiess, H.: A maximum likelihood model for estimating origin-destination matrices. Transportation Research B **21**(5) (1987) 395–412
15. Cascetta, E., Nguyen, S.: A unified framework for estimating or updating origin/destination matrices from traffic counts. Transportation Research B **22**(6) (1988) 437–455
16. Cascetta, E.: Estimation of trip matrices from traffic counts and survey data: A generalized least squares estimator. Transportation Research B **18**(4/5) (1984) 289–299
17. Bell, M.: The estimation of origin-destination matrices by constrained generalized least squares. Transportation Research B **25**(1) (1991) 13–22
18. Rossi, F., van Beek, P., Walsh, T.: Handbook of Constraint Programming (Foundations of Artificial Intelligence). Elsevier Science Inc. (2006)
19. Siikonen, M.L., Susi, T., Hakonen, H.: Passenger traffic flow simulation in tall buildings. Elevator World **49** (2001) 117–123
20. Strakosch, G.: Vertical Transportation: Elevators and Escalators. John Wiley & Sons (1983)
21. Barney, G.: Elevator Traffic Handbook: Theory and Practice. Spon Press (2003)